

Efficient RDMA-based Multi-port Collectives on Multi-rail QsNet^{II} Clusters

Ying Qian and Ahmad Afsahi

Department of Electrical and Computer Engineering

Queen's University

Kingston, ON, Canada



- Introduction
- Overview of QsNet^{II}
- Experimental Framework
- Motivation
- Collective Algorithms
- Performance Results
- Conclusions



- Clusters have become the predominant computing platforms providing **high-performance**, and **high-availability**. Such systems:
 - Tackle high-end applications known as **Grand Challenge** problems
 - Provide services such as search engines, data mining, eCommerce, web hosting, digital libraries
- Main components in **Clusters of Multiprocessors**:
 - The Symmetric Multiprocessors (SMP) node
 - The Communication Subsystem
- Clusters need high-performance networks, and efficient communication system software.
 - Myrinet
 - QsNet^{II}
 - InfiniBand



- Multi-rail networks have been introduced to overcome bandwidth limitations for today's most demanding applications, and to enhance fault tolerance.
 - QsNet^{II} (native support) and InfiniBand (MVAPICH)
- How to distribute the traffic over multiple rails?
 - Multiplexing
 - Message Striping
- QsNet^{II} uses multiple NICs per node to construct a multi-rail network.
 - Native support at the user level for a simple **even message striping**
 - Only for **point-to-point** messages



- Many parallel applications need communication patterns that involve collective data movement and global control, known as **collective communications**.
- Such applications mostly use **Message Passing Interface (MPI)**.
 - MPI supports both point-to-point and collective communications.
 - Efficient and scalable implementation of collectives is very important to the performance of such applications.
- **Problem statement:**
 - Can we design and implement efficient collectives on top of multi-rail Quadrics QsNet^{II}?
 - What are the design challenges?
 - How much performance improvement can be achieved?



- Introduction
- Overview of QsNet^{II}
- Experimental Framework
- Motivation
- Collective Algorithms
- Performance Results
- Conclusions



- QsNet^{II} is the latest generation interconnect from Quadrics, offering low latency and high bandwidth. It consists of two ASICs: Elan4 and Elite4.
 - Provides a protected user-level access to NIC.
 - Supports a globally shared virtual-memory system. Data can be transferred directly from a source virtual address to a destination virtual address.
 - Supports a connectionless model, with reliable and ordered delivery.
 - Communication software includes both MPI and SHMEM, implemented with libelan and libelan4 in the Elan library.



- QsNet^{II}:

- Supports point-to-point **send/receive**, and **Remote Direct Memory Access (RDMA)** models.
 - ❖ RDMA write (*elan_put*) and RDMA read (*elan_get*) are supported. **No need to register buffers.**
 - ❖ Send/receive mode is supported through **Tports**, a two-sided message-passing semantic as in MPI. MPI uses Tports as its transport layer.
 - ❖ Even message striping support for large messages on multi-rail networks for Elan RDMA put and get, SHMEM put and get, and Tports send/receive.



- QsNet^{II} supports a number of collectives directly at the Elan level.
 - Hardware broadcast ([elan_hbcast](#)), and hardware barrier ([elan_gsync](#))
 - Reduction ([elan_reduce](#)), gather ([elan_gather](#)), and personalized all-to-all ([elan-alltoall](#))
 - ❖ These collectives are directly used by the MPI library. Enhancing their performance will directly affect the MPI layer.
 - Collectives are currently implemented on top of point-to-point Tports or `elan_put`, and may gain multi-rail striping from those underlying subsystems.
 - ❖ *elan_put* and *elan_doput* are non-blocking functions used in the implementation.
 - ❖ *elan_doput* is similar to *elan_put* with the exception that it sets a destination event on completion.
 - ❖ *elan_wait* is used to test or wait on for completion of the transfer.



- Introduction
- Overview of QsNet^{II}
- Experimental Framework
- Motivation
- Collective Algorithms
- Performance Results
- Conclusions

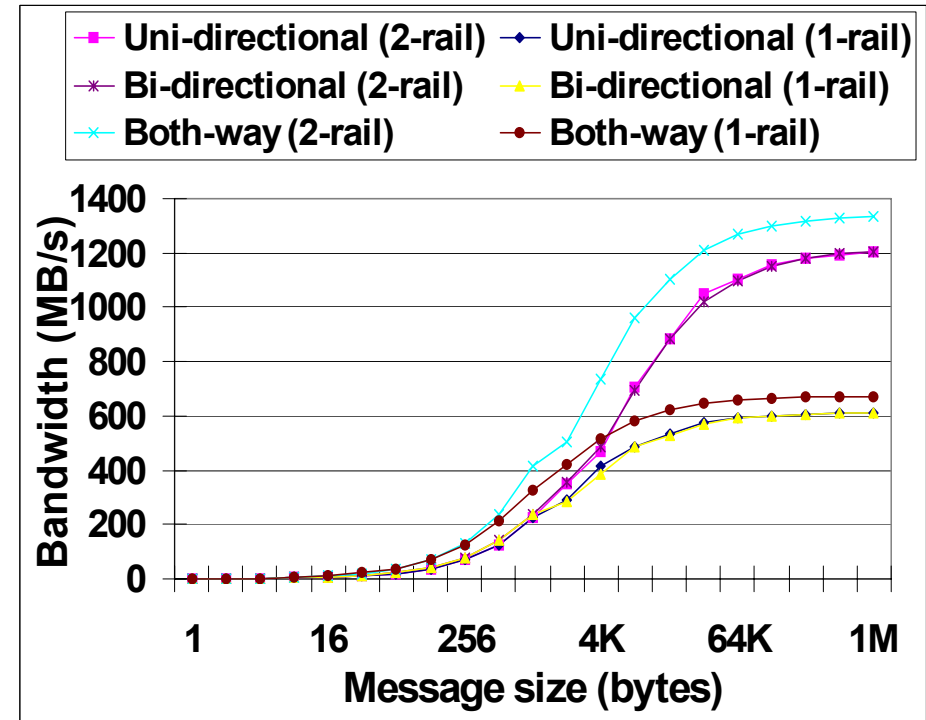
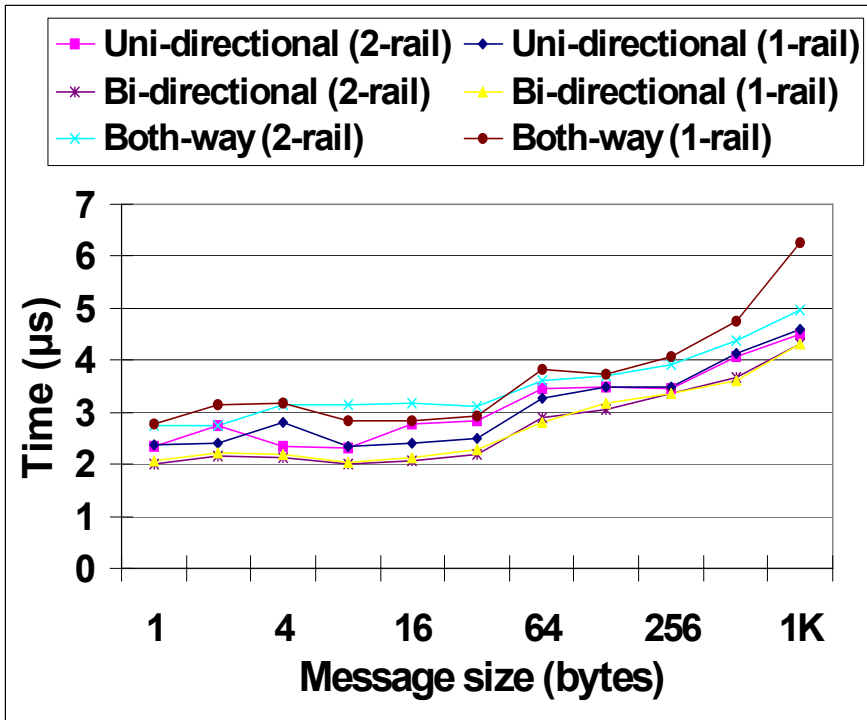


	Four 4-way Dell PowerEdge 6650
Processors	4 – 1.4 GHz Intel Xeon MP per node
Kernel	Vanilla version 2.6.9
L1 cache	12KB execution trace cache, 8KB data cache
L2 cache	256KB shared and unified
L3 cache	512KB shared and unified, cache
Main Memory	2GB DDR-SDRAM per node, on a 400 MHz FSB
QsNet ^{II} switch	Two QS8A-AA E-series 8-way switches
QsNet ^{II} NIC	Two QM500-B per node
Quadrics software	Hawk release with the kernel patch qsnetp2, kernel module 5.10.qsnet, QsNet library 1.5.6-1, and QsNet ^{II} library 2.2.4-1.
Launch tool	pdsh 2.6.1
MPI implementation	Quadrics MPI, version 1.24-47.intel81
PCI-X	100 MHz, 64 bit



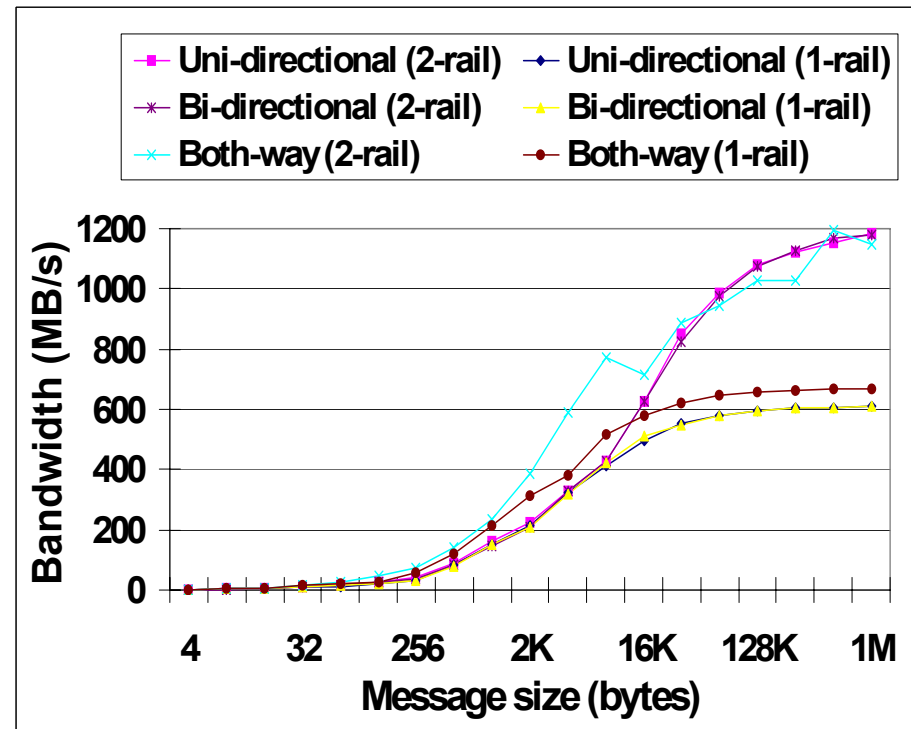
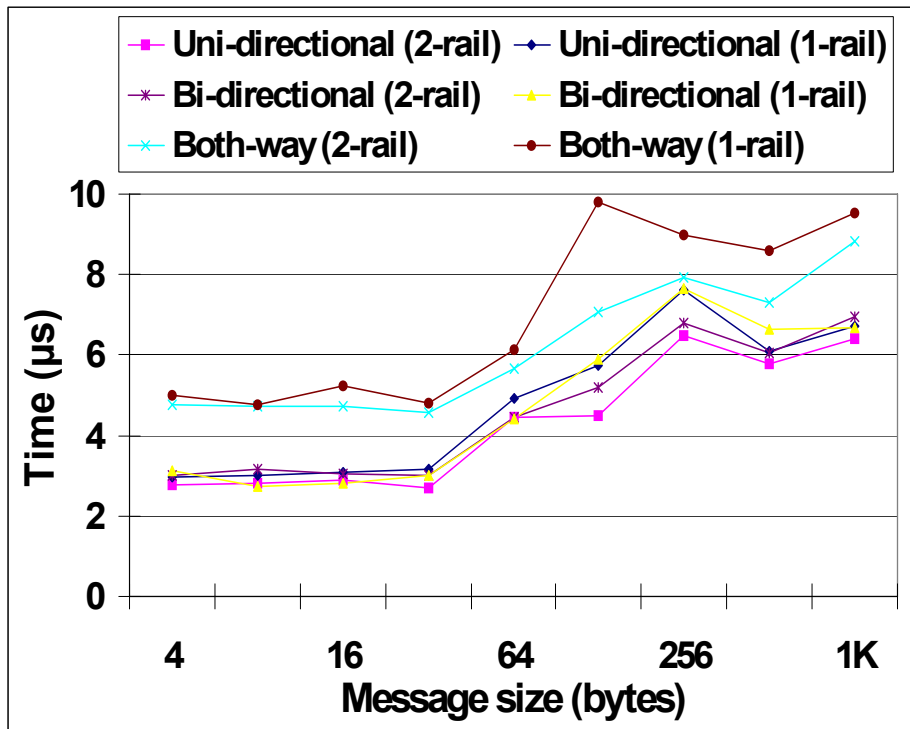
- Introduction
- Overview of QsNet^{II}
- Experimental Framework
- **Motivation**
- Collective Algorithms
- Performance Results
- Conclusions





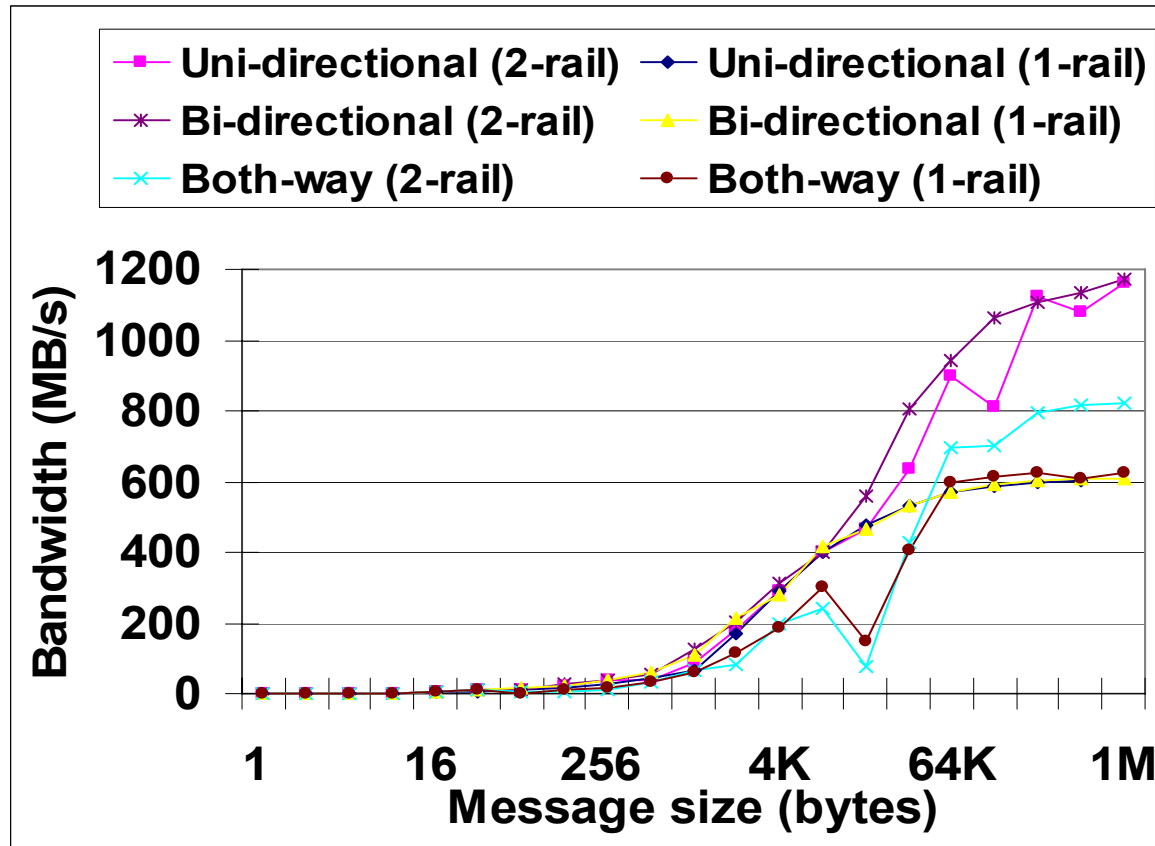
- Two-rail has a much better performance than single-rail.
- *elan_get* achieves the same bandwidth, however, has a slightly larger short message latency.





- The single-rail bandwidth is similar to RDMA, but dual-rail falls short of.
- The short message latency of Tports is slightly larger than the RDMA.

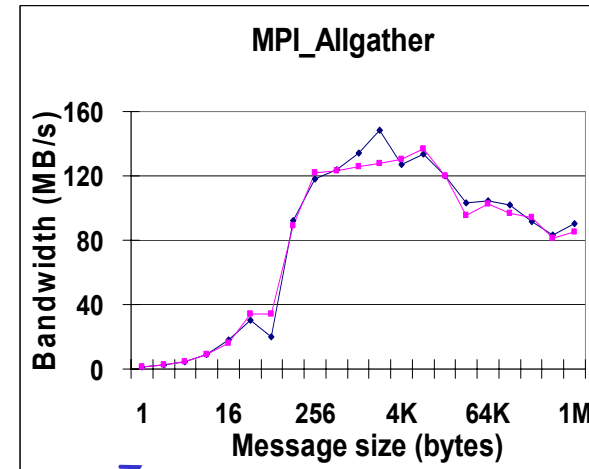
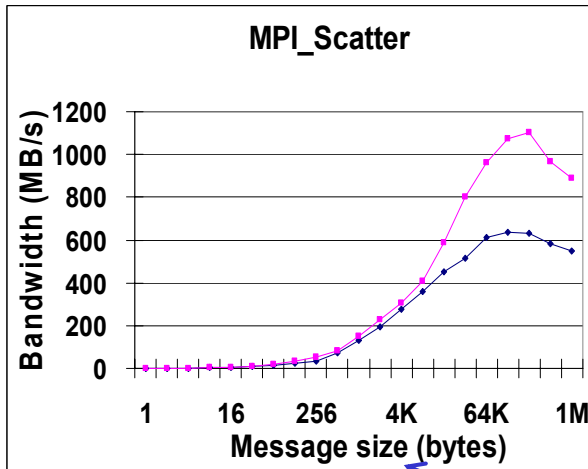
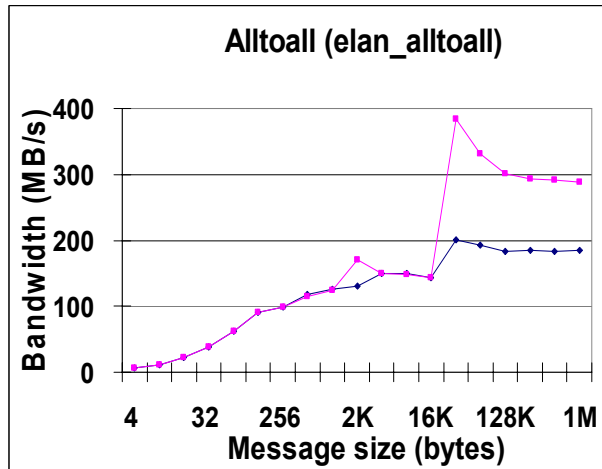
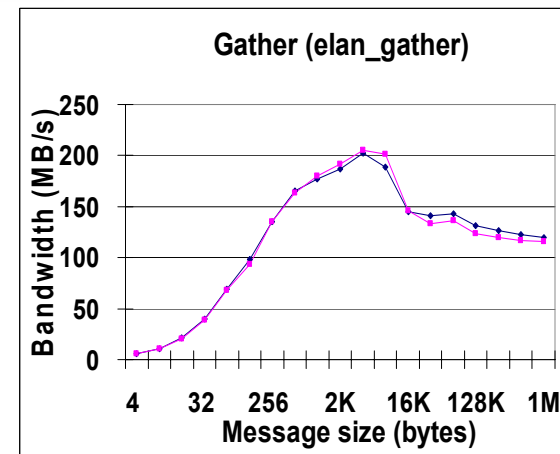
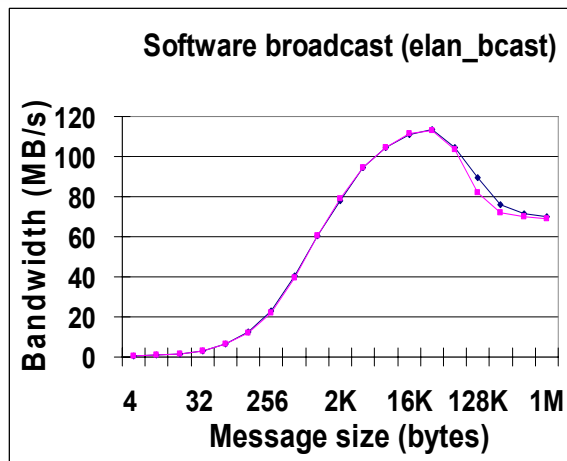
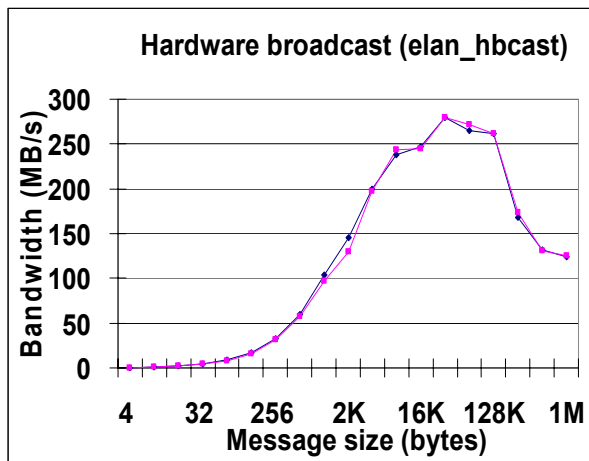




- Both-way MPI bandwidth is not affected by the dual-rail.
- The short message MPI latency is close to that of the Tports.



◆ Single-rail ■ Dual-rail



On top of Tports



- With a few exceptions, the current collectives do not benefit from the multi-rail systems. Excellent opportunities exist for devising multi-rail collectives. Those include:
 - Designing single-port collective algorithms that gain multi-rail striping from the underlying communication subsystem.
 - Designing multi-port collective algorithms for multi-rail systems that also benefit from multi-rail striping.
- ❖ We will focus on **Scatter**, **Gather**, and **All-to-all Personalized Exchange** collectives, and implement them directly at the Elan level.



- Introduction
- Overview of QsNet^{II}
- Experimental Framework
- Motivation
- Collective Algorithms
- Performance Results
- Conclusions



- Assume N is the number of processes, and k is the number of ports in the multi-port algorithms (equal to the number of available rails).
- Each process has the ability to send and receive k messages simultaneously on its k links.
- The assumption is that the number of processes, N , is a power of $(k + 1)$. Otherwise, dummy processes is assumed to exist until the next power of $(k + 1)$, and the algorithms apply with little or no performance loss.
- **Definitions:**
 - **Scatter operation:** the root process has a unique message for each of the remaining $N - 1$ processes.
 - **Gather operation:** is the exact reverse of the scatter operation. The root process receives a message from each of the remaining $N - 1$ processes.
 - **All-to-all Personalized Exchange operation:** in this collective, each process has a unique message for each of the remaining $N - 1$ processes.



- Algorithm A: Multi-port Spanning Binomial Tree algorithm for Scatter
 1. The scattering process sends k messages of length $N/(k + 1)$ each to its k children. At the end of this step, there are $(k + 1)$ processes having $N/(k + 1)$ different messages.
 2. These processes at step 2 send one $(k + 1)$ -th of their initial message to each of their immediate k children.
 3. This process continues and all processes are informed after $\log_{k+1} N$ steps.

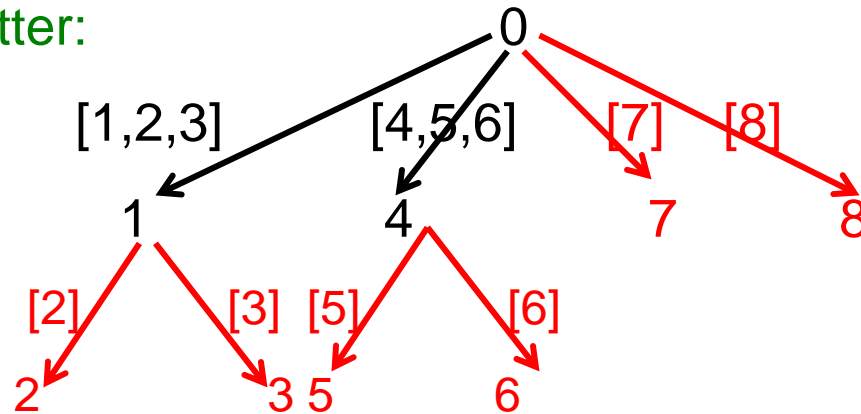
$$T = (t_s \times \log_{k+1} N) + (l_m \times \tau) \sum_{i=1}^{\log_{k+1} N} (k + 1)^{(\log_{k+1} N) - i}$$

$$T = (t_s \times \log_{k+1} N) + \frac{N - 1}{k} (l_m \times \tau)$$

t_s is the message startup time, l_m is the message size in bytes, and τ is the time to transfer one byte.



- 9-process scatter:



- This algorithm has a logarithmic number of steps and suitable for short messages.
- Processes may be spanned such that packing/unpacking would not be needed. Otherwise, intermediate copies may be needed.



- Algorithm B: Multi-port Sequential Tree algorithm for Scatter:
 - At each step, the scattering process sends its k different messages to k other processes. There are a total of $(N - 1)/k$ communication steps. Therefore, the total communication cost is:

$$T = \frac{N - 1}{k} \times (t_s + l_m \times \tau)$$

- Suitable for medium to large message sizes



- Multi-port Spanning Binomial Tree algorithm for Gather:
 - The algorithm is the exact reverse of the scatter, and the same spanning tree can be used.
 - Communication starts from the leaf processes.
 - Messages are combined in the intermediate processes until it reaches the gathering root.
 - The total communication cost is:

$$T = (t_s \times \log_{k+1} N) + \frac{N-1}{k} (l_m \times \tau)$$



- Multi-port Direct algorithm for All-to-all Personalized Exchange:
 - Processes are arranged in a virtual ring.
 - At step i , process p sends its message to processes $(p + (i - 1)k + 1) \bmod N, (p + (i - 1)k + 2) \bmod N, \dots, (p + ik) \bmod N$.
 - The total communication cost is:

$$T = \frac{N - 1}{k} \times (t_s + l_m \times \tau)$$

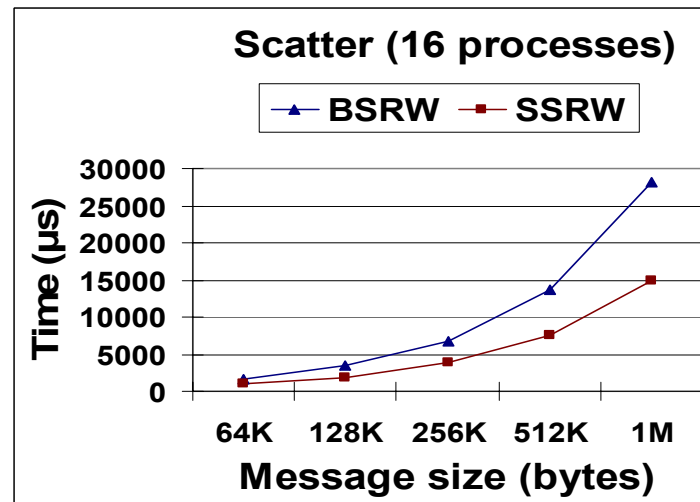
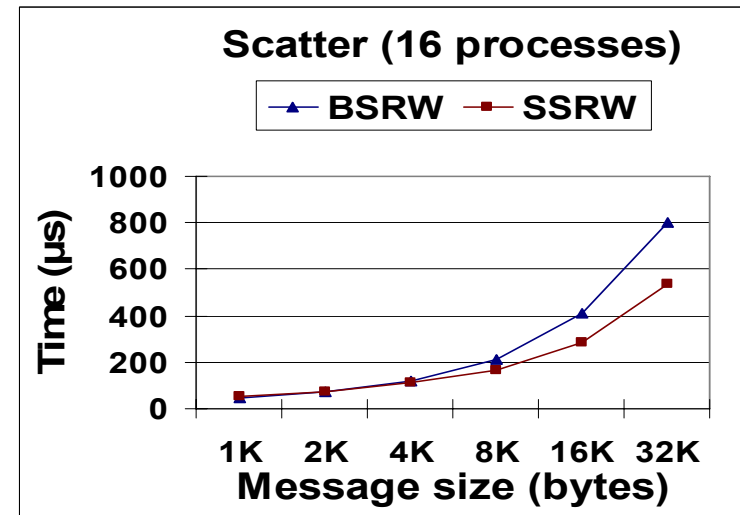
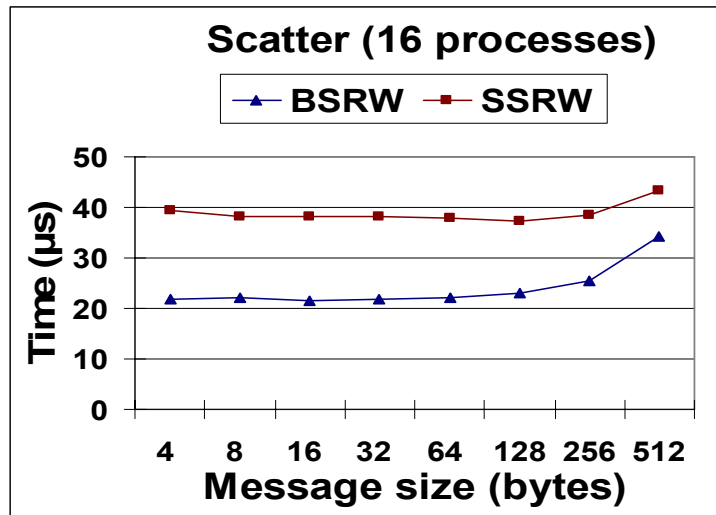


- No need for memory registrations
- No need for address exchange for message transfers.
- Senders have direct control over the rails by using `elan_doput` function. Even message striping is used.
 - `elan_wait` is used to make sure the user-buffer can be re-used.
- Remote event notification is enabled in `elan_doput` for multi-rail systems. This allows multi-rail striped (`ELAN_RAIL_ALL`) put messages to have a `devent` (destination event) for each rail.
 - The destination event is set once in each rail and the destination process calls `elan_initEvent` once for each rail, and wait for each `ELAN-EVENT` to be returned. This guarantees a message is delivered in its entirety on a multi-rail system.
- No synchronization is used in the implementations.



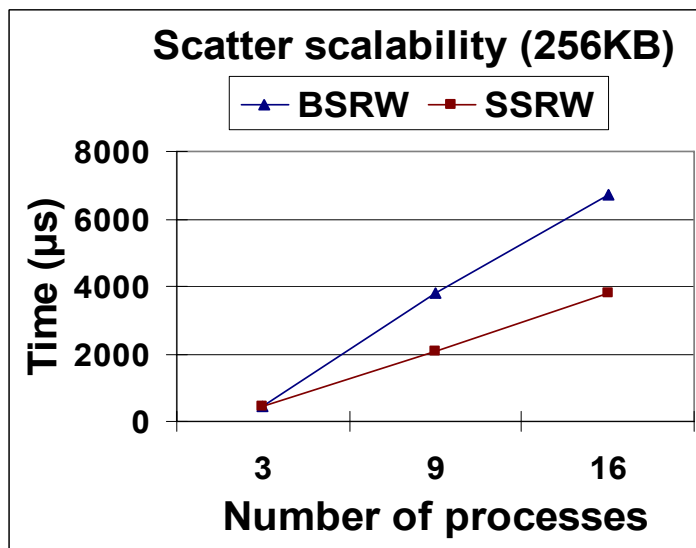
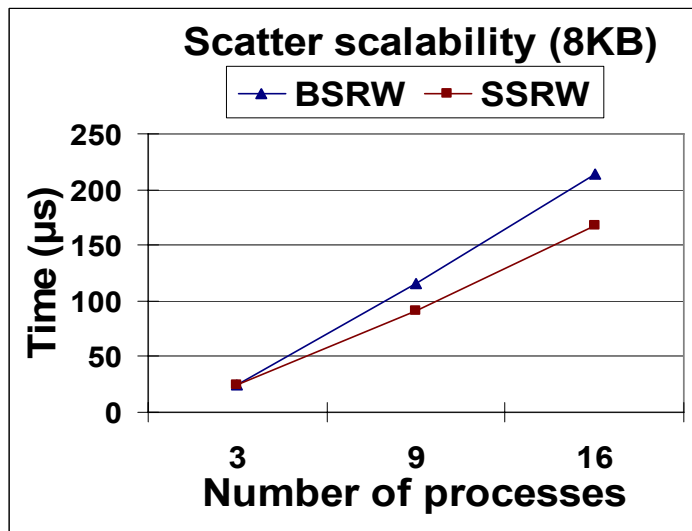
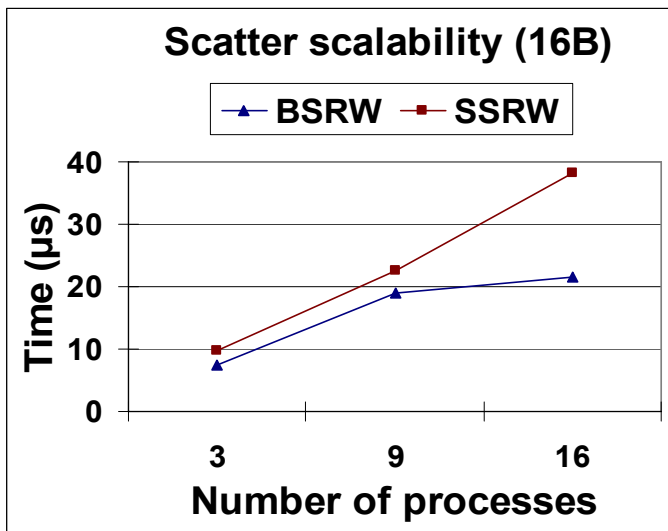
- Introduction
- Overview of QsNet^{II}
- Experimental Framework
- Motivation
- Collective Algorithms
- Performance Results
- Conclusions

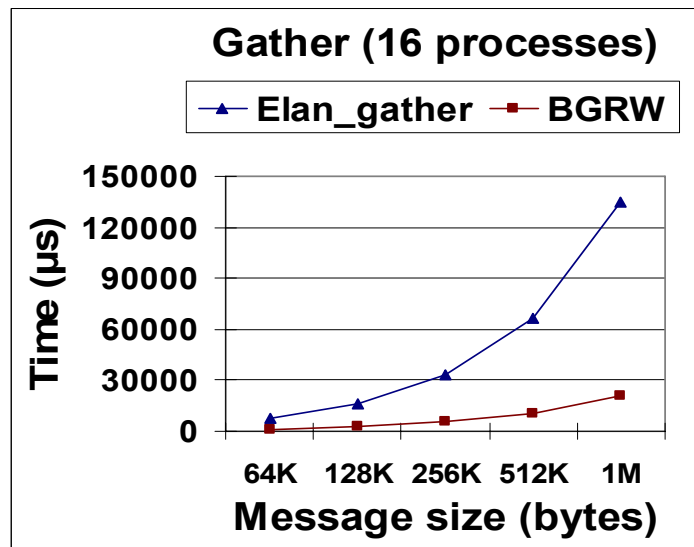
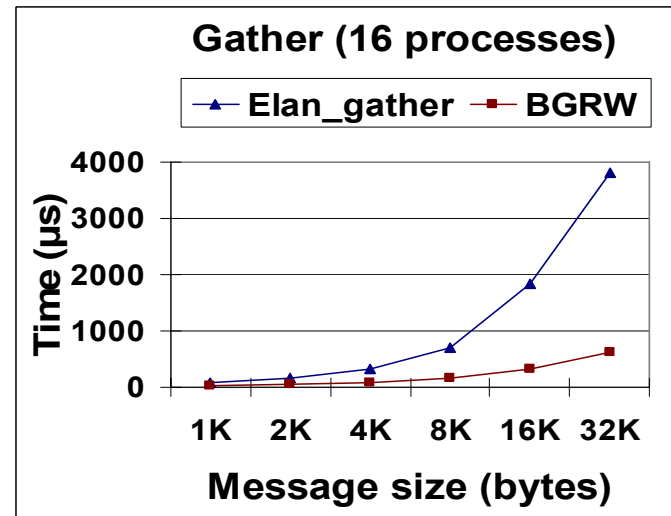
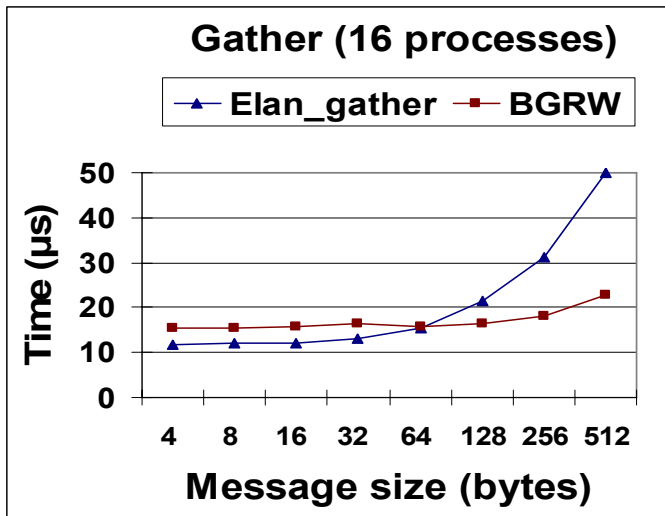




As expected, the multi-port binomial algorithm is better for short messages.

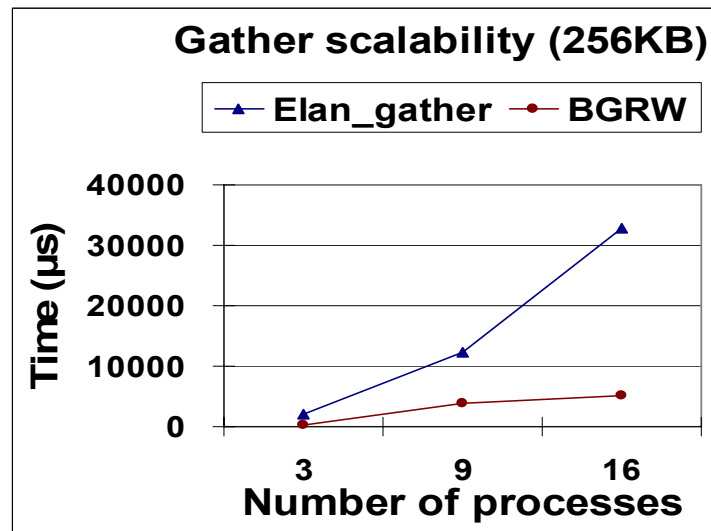
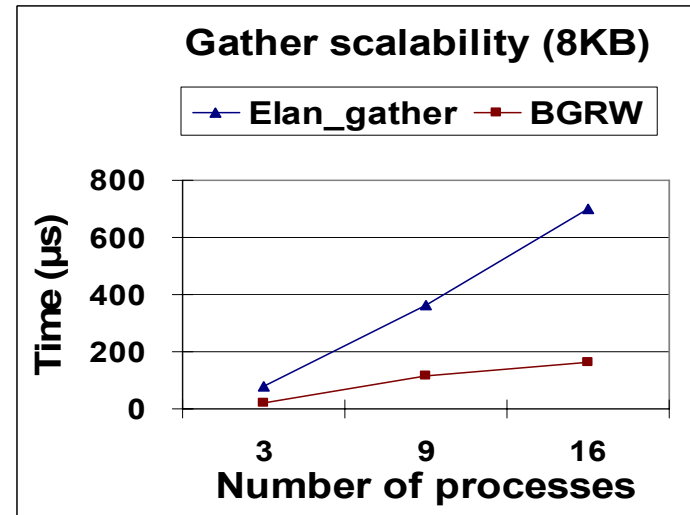
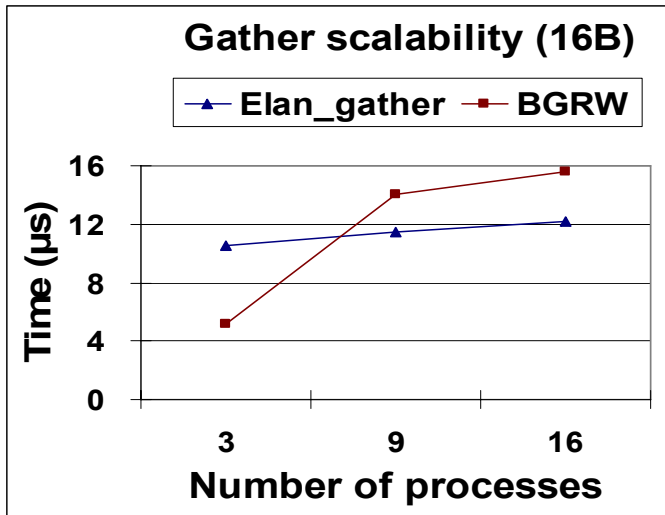






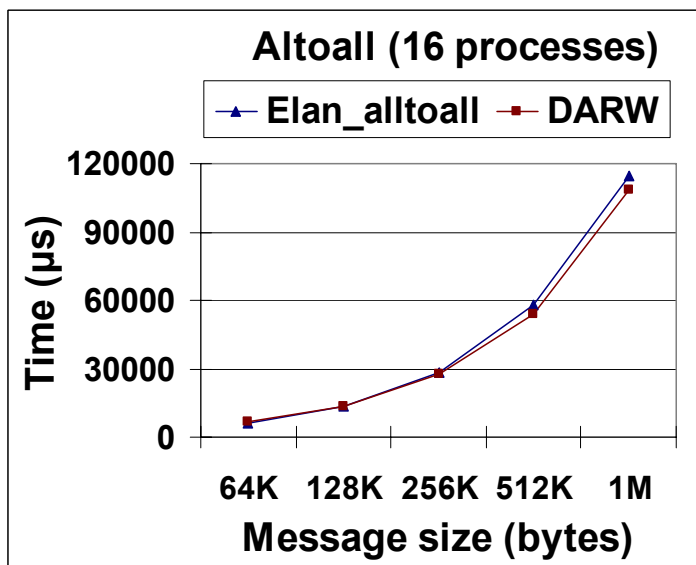
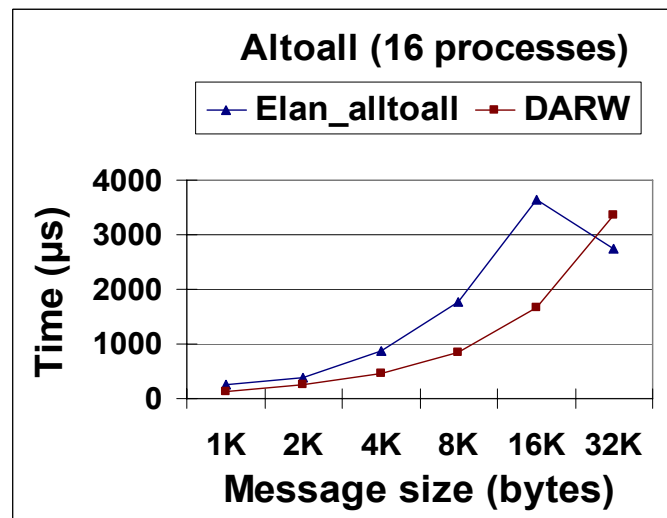
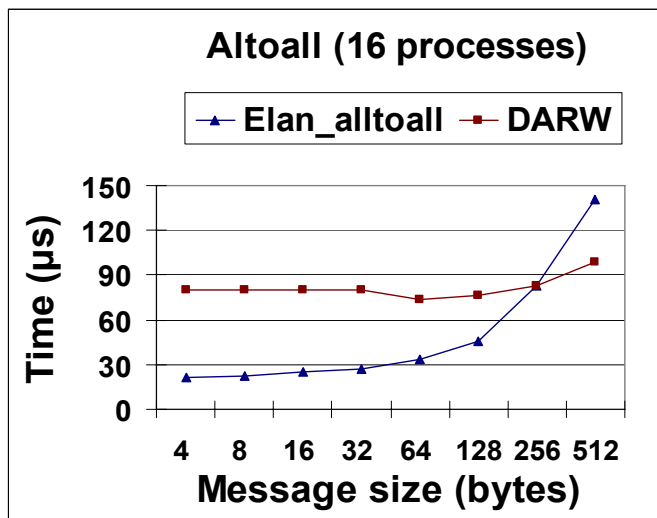
Improvement of up to 6.35 for 1MB message.





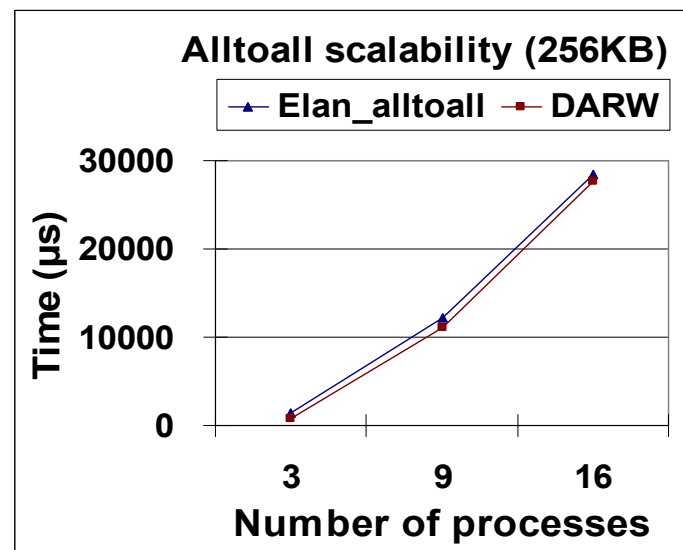
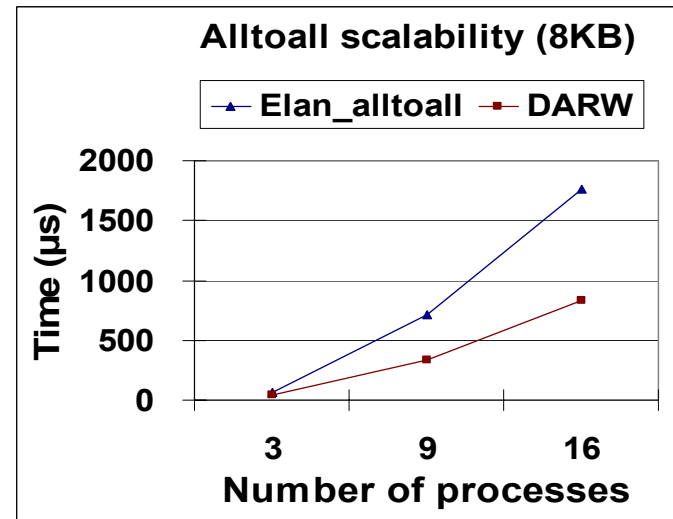
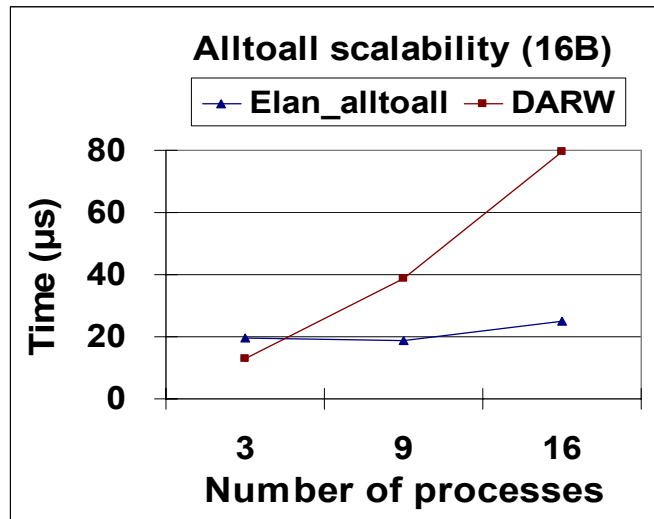
With increasing number of processes, elan_gather is better for short messages.





Improvement of up to 2.19 for 16KB message.





- Introduction
- Overview of QsNet^{II}
- Experimental Framework
- Motivation
- Collective Algorithms
- Performance Results
- Conclusions



- Efficient implementation of collective operations is critical to the performance of applications.
- We proposed a couple of **multi-port** algorithms for short and long messages for the **scatter** operation.
- The implementation of the **multi-port gather** gains an improvement of up to 6.35 for 1MB message over *elan-gather*.
- The **multi-port all-to-all personalized exchange** performs better than *elan_alltoall* by a factor of 2.19 for 16KB messages.
- For short messages, better algorithms should be used.



- Work on **standard exchange**, and **bruck's index** algorithms are under way.
- We intend to use shared memory for short messages among the co-located processes to speedup the collectives.
- Adaptive striping will also be considered.
- We will be working on other collectives as well.
- NIC-based or NIC-assisted collectives for multi-rail networks are of interest.
- Basic hardware collectives could be used to implement other collectives.
- Evaluate using larger testbeds.



- This work was supported by:
 - Natural Sciences and Engineering Research Council of Canada (NSERC)
 - Canada Foundation for Innovation (CFI)
 - Ontario Innovation Trust (OIT)
 - Queen's University
 - Ontario Graduate Scholarship for Science and Technology (OGSST)



